

Ferramenta para auxílio na análise de impacto e rastreabilidade de requisitos na gestão de mudanças

Aluno: José Alberto Zimmermann

Orientador: Marcel Hugo

Banca: Everaldo Artur Grahl

Joyce Martins

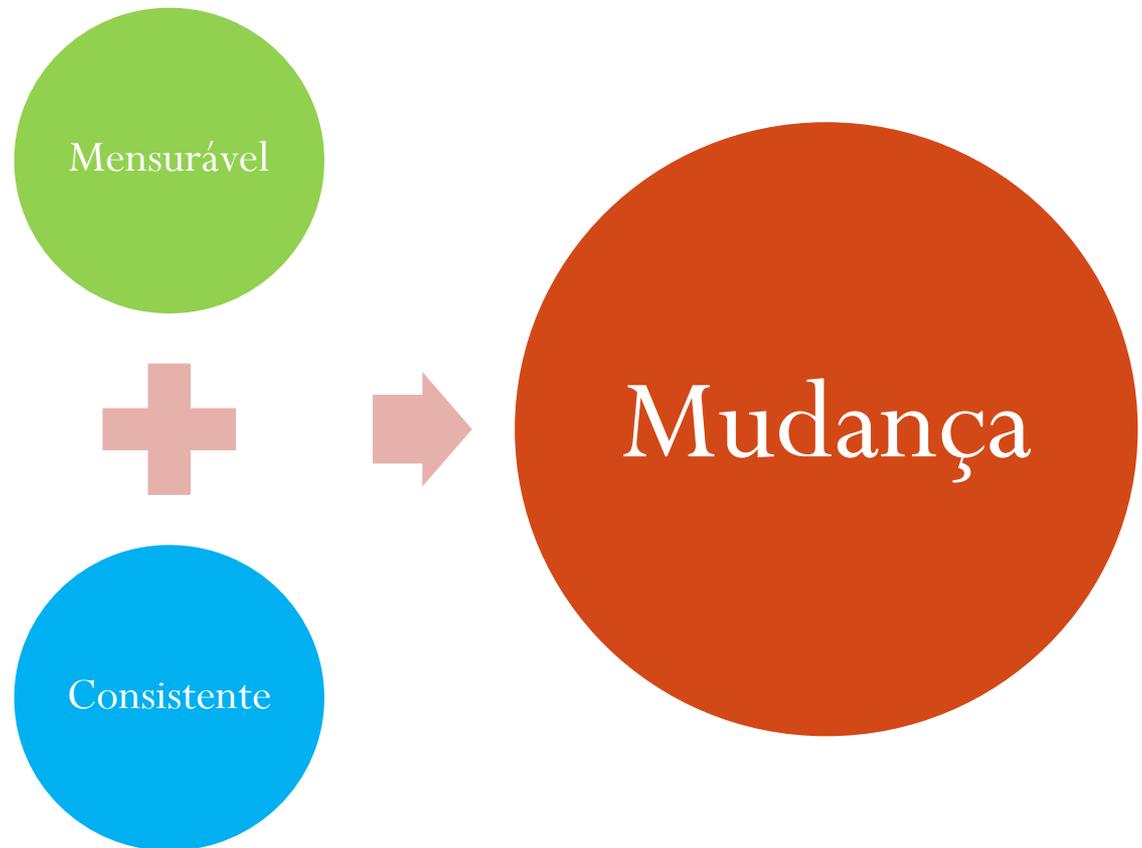
Roteiro

- Introdução
 - Contexto do problema
 - Objetivos gerais e específicos
- Principais conceitos e fundamentação
 - Trabalhos relacionados
- Desenvolvimento do Trabalho
 - Requisitos
 - Especificação da solução
 - Implementação
 - Protótipo
 - Resultados e discussões
- Conclusão e extensões

Introdução

Contexto do problema

- Necessidade de se garantir que as alterações sejam:
 - Consistentes
 - Mensuráveis



Contexto do problema

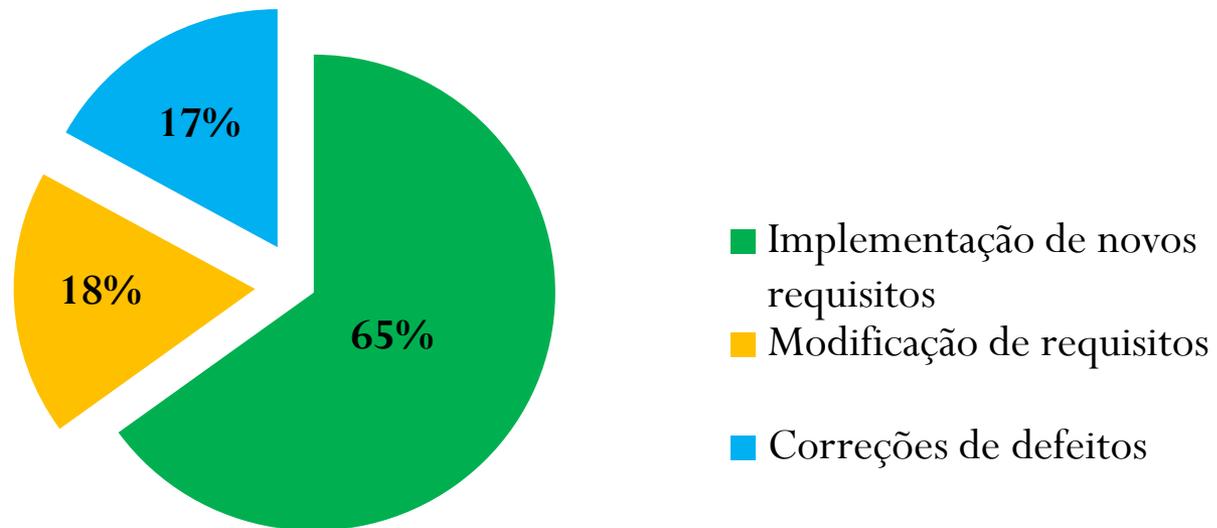
- Empresas da área de informática possuem dificuldades em gerir o processo de mudança

Velocidade

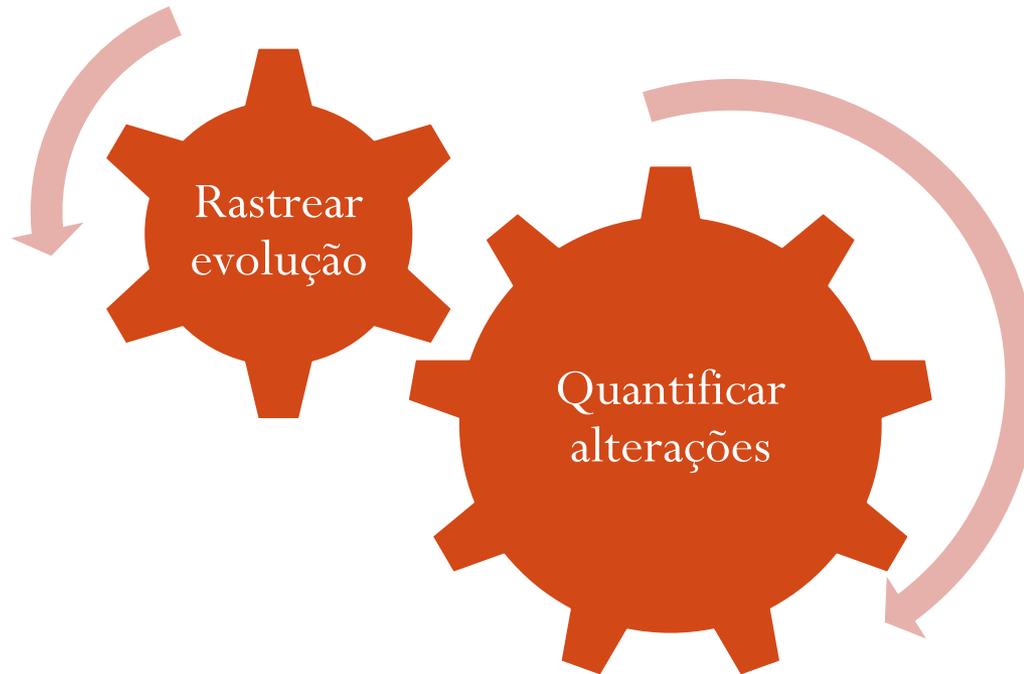
Modificação de requisitos

Contexto do problema

- Segundo Sommerville (2003, p. 518):



Contexto do problema

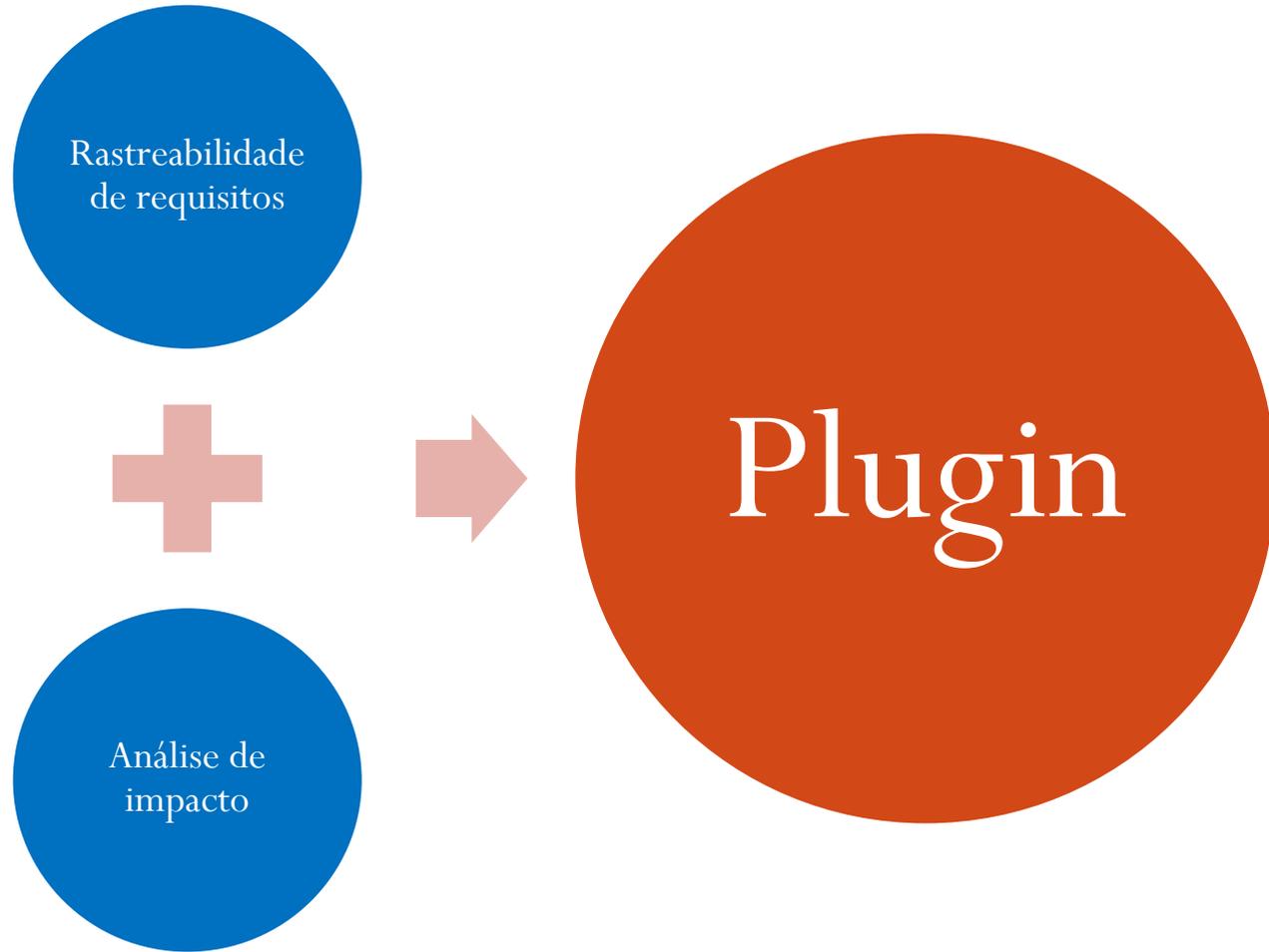


Objetivos

Objetivo principal

Objetivos específicos

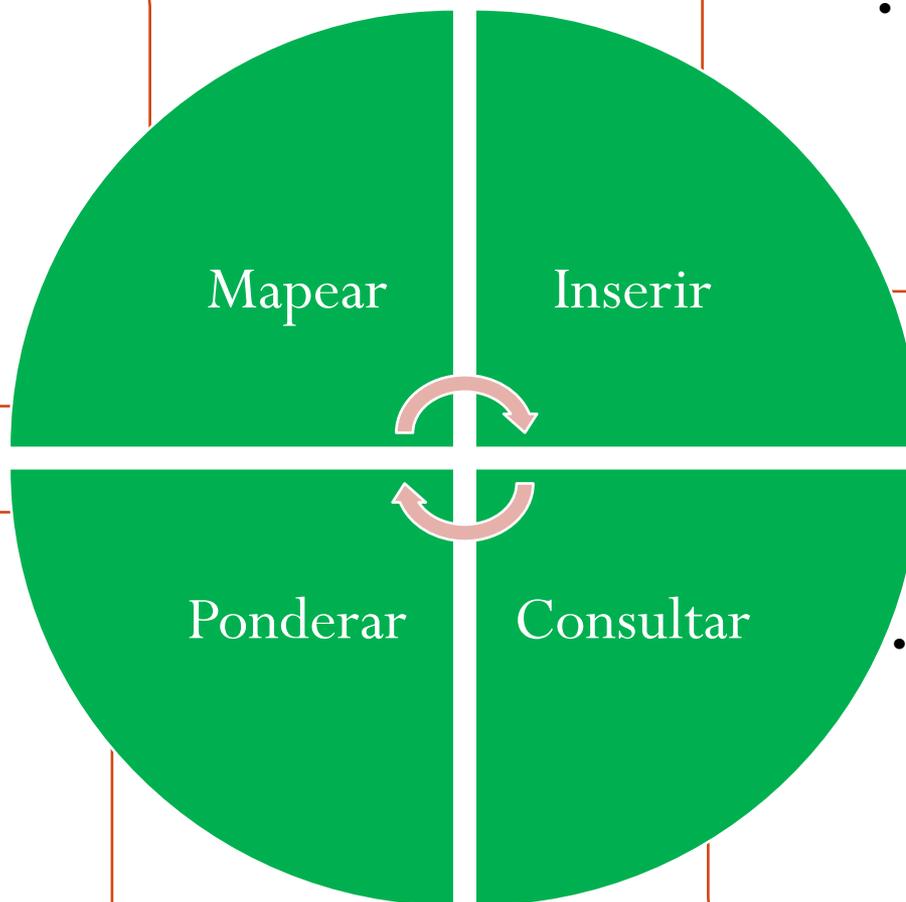
Objetivo principal



Objetivos específicos

- Efetuar o mapeamento dos requisitos de softwares nos códigos fonte do projeto

- Definir conjunto de anotações que permita ao usuário inserir anotações



- Ponderar os custos das alterações dos requisitos, utilizando o AHP

- Interface que permita a análise de impacto relativo às alterações

Principais conceitos e fundamentação

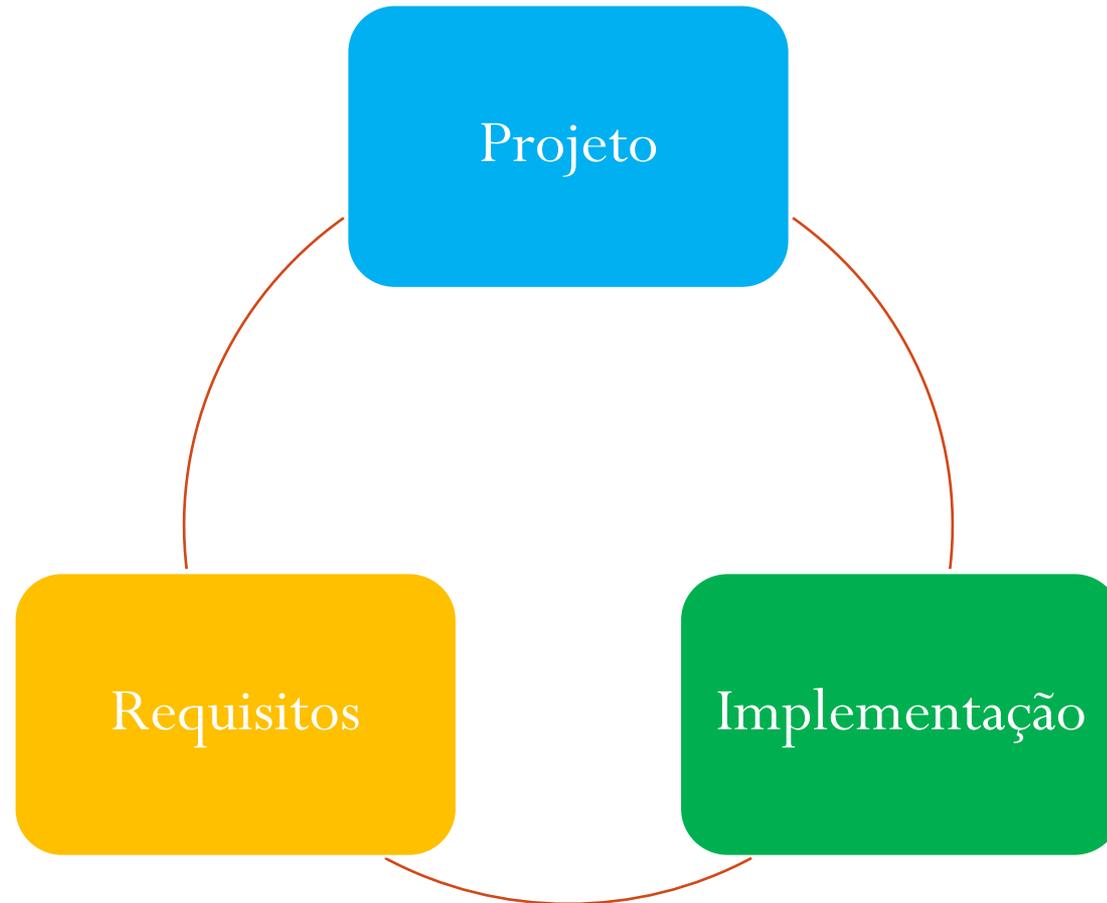
Rastreabilidade de requisitos

Metodologia AHP

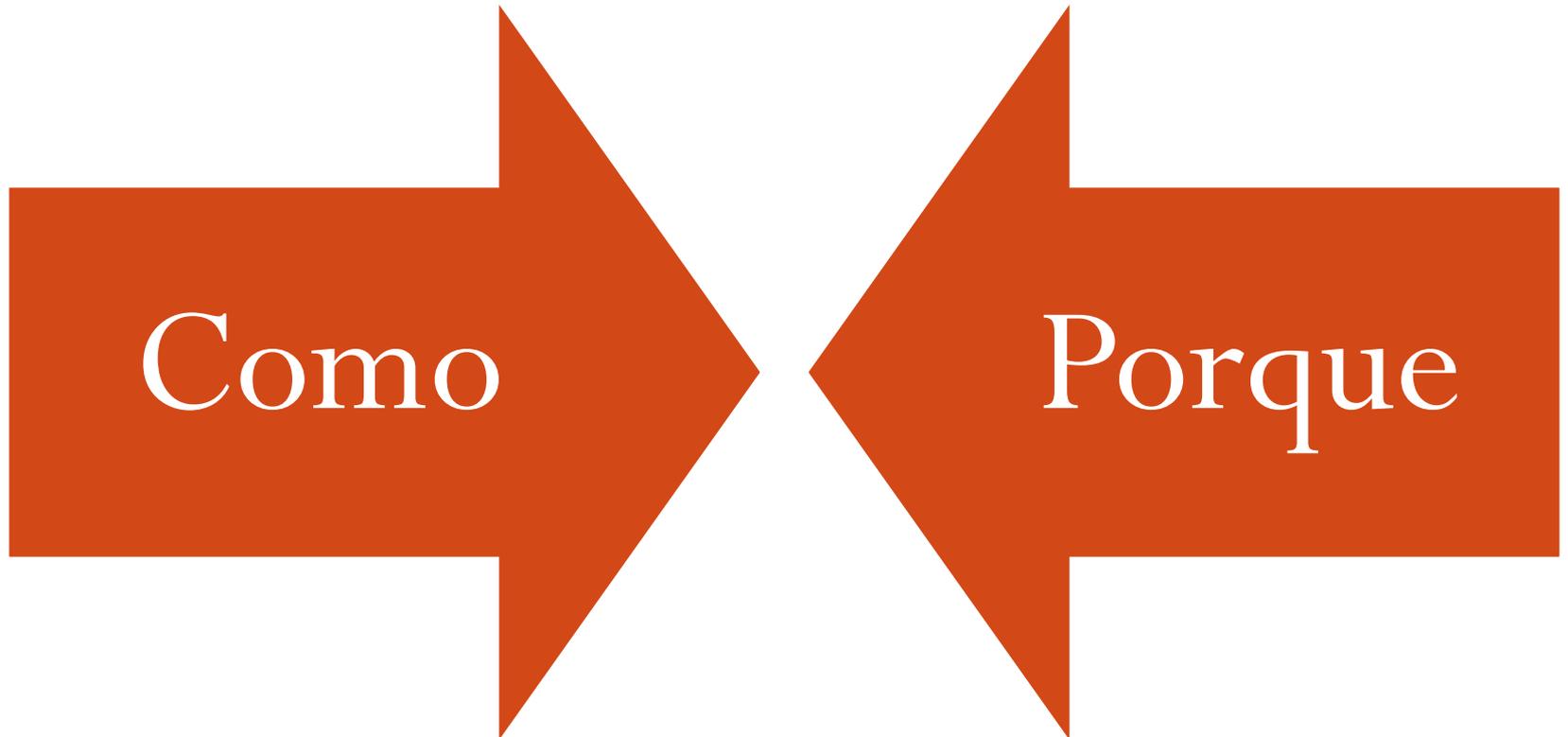
Eclipse

Enterprise Architect

Rastreabilidade de requisitos

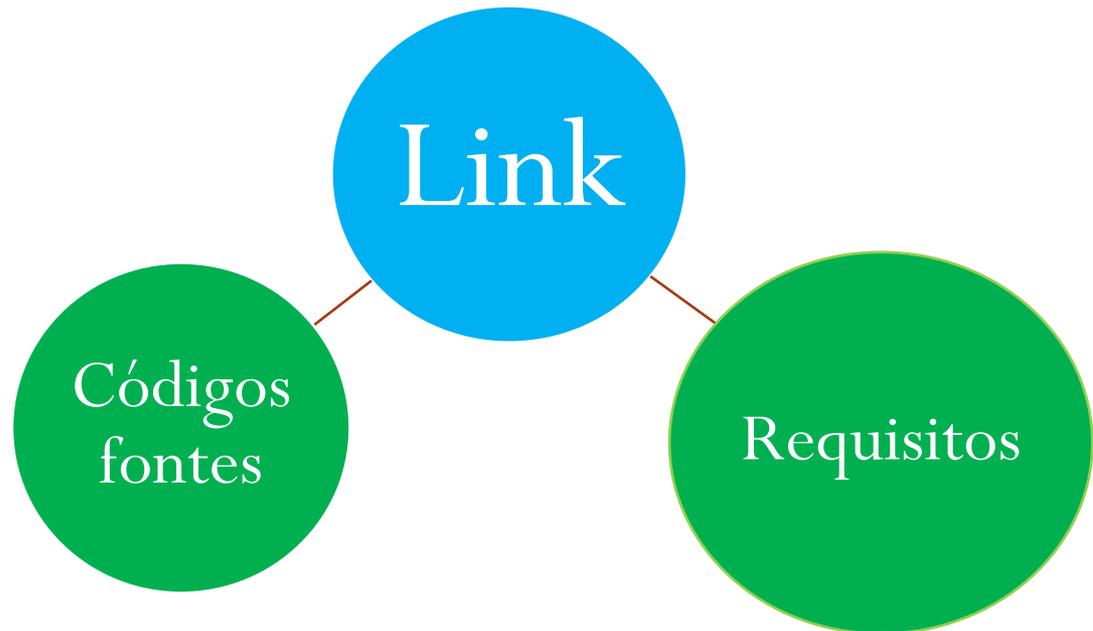


Rastreabilidad de requisitos



Rastreabilidade de requisitos

- A elaboração de projetos de software deve produzir requisitos rastreáveis
- Permite que as estimativas de custos de alterações sejam mais precisas

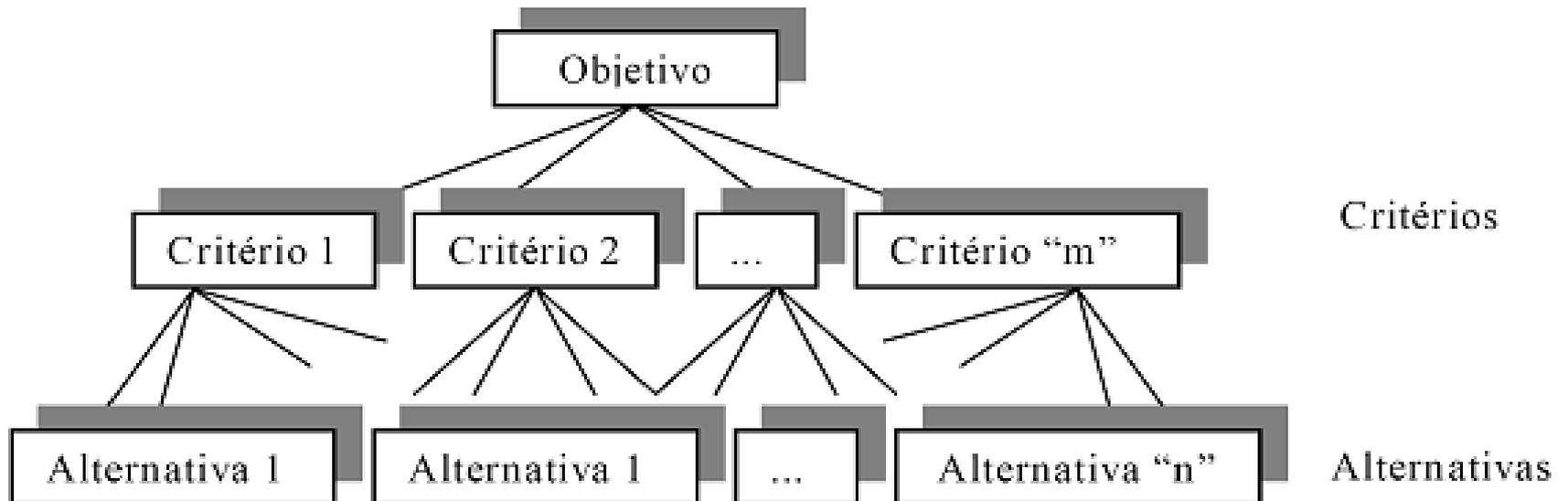


Metodologia AHP

- Ponderar as características qualitativas
- Permite a ponderação e priorização de cada um dos requisitos de um projeto
- Permite obter o valor percentual de cada requisito, em relação a um projeto
- Trata-se de um método que decompõe o problema em vários níveis, quebrando um problema em fatores

Metodologia AHP

- O pensamento analítico consiste:



Metodologia AHP

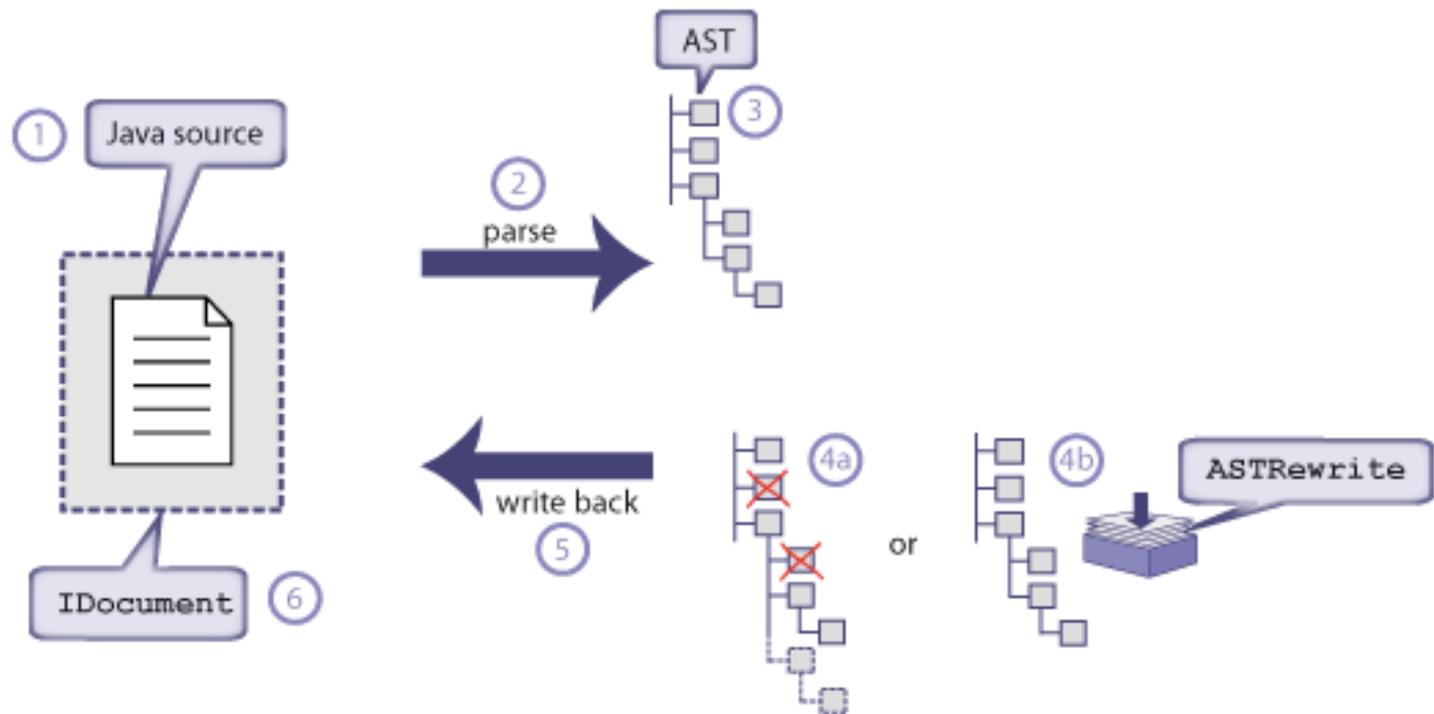
Escala	Definição
1	Menos importante
3	Importância pequena
	Importância grande ou essencial
5	Importância muito grande
7	Importância absoluta
2,4,6,8	Valores intermediários. Se na atividade “j” recebe um dos valores acima, quando comparada com a atividade “i”, então “j” tem o mesmo valor recíproco de “i”
Racionais	Razão da escala

Eclipse

- Plataforma de desenvolvimento concebida para ser extensível
- Permite integrar janelas de menu, views e outras interfaces
- É desenvolvido utilizando a biblioteca SWT

Eclipse

- AST – Fluxo de funcionamento

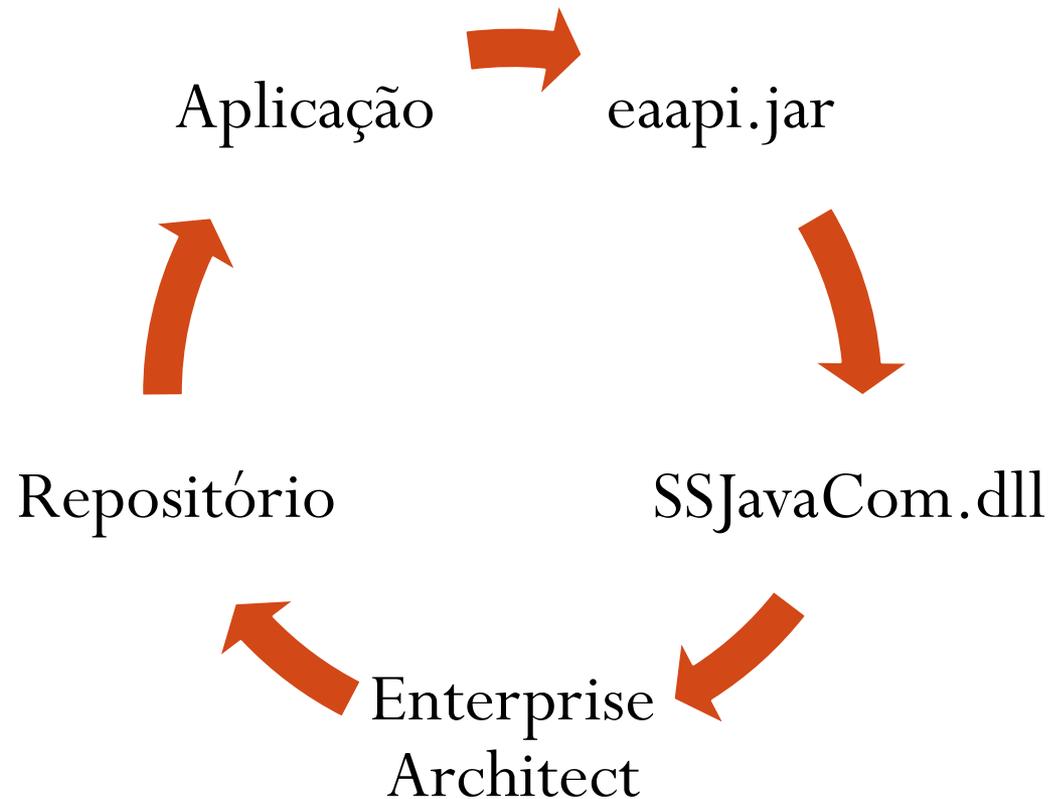


Enterprise Architect

- Ferramenta CASE utilizada para modelagem e construção de projetos de sistemas de software
- Utiliza a UML para descrever de maneira visual os elementos de um projeto
- Criação de links entre os artefatos do projeto

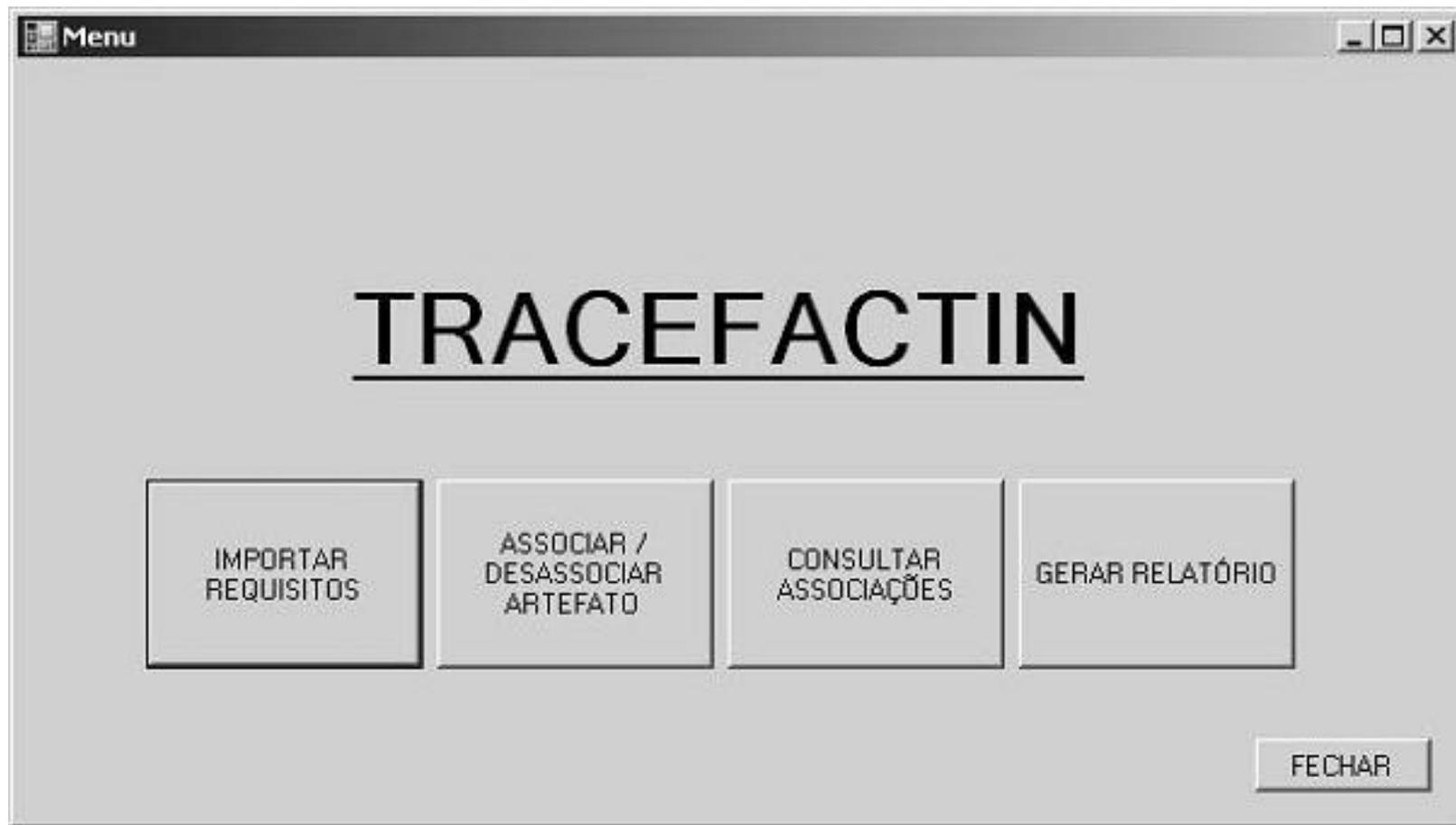
Enterprise Architect

- Integração das funcionalidades da ferramenta EA com o Java



Trabalhos relacionados

TraceFact-In



IRequirement

The screenshot displays the 'Menu Principal - IRequirement' application window. The interface includes a menu bar with 'Arquivo', 'Exibir', 'Configurações', 'Relatórios', and 'Ajuda'. A toolbar contains various icons for file operations and system settings. The main area is divided into a left sidebar and a central grid.

The left sidebar shows a tree view with the following structure:

- Documento ERS
 - Rastreabilidade
 - RF x RNF
 - RF x CSU

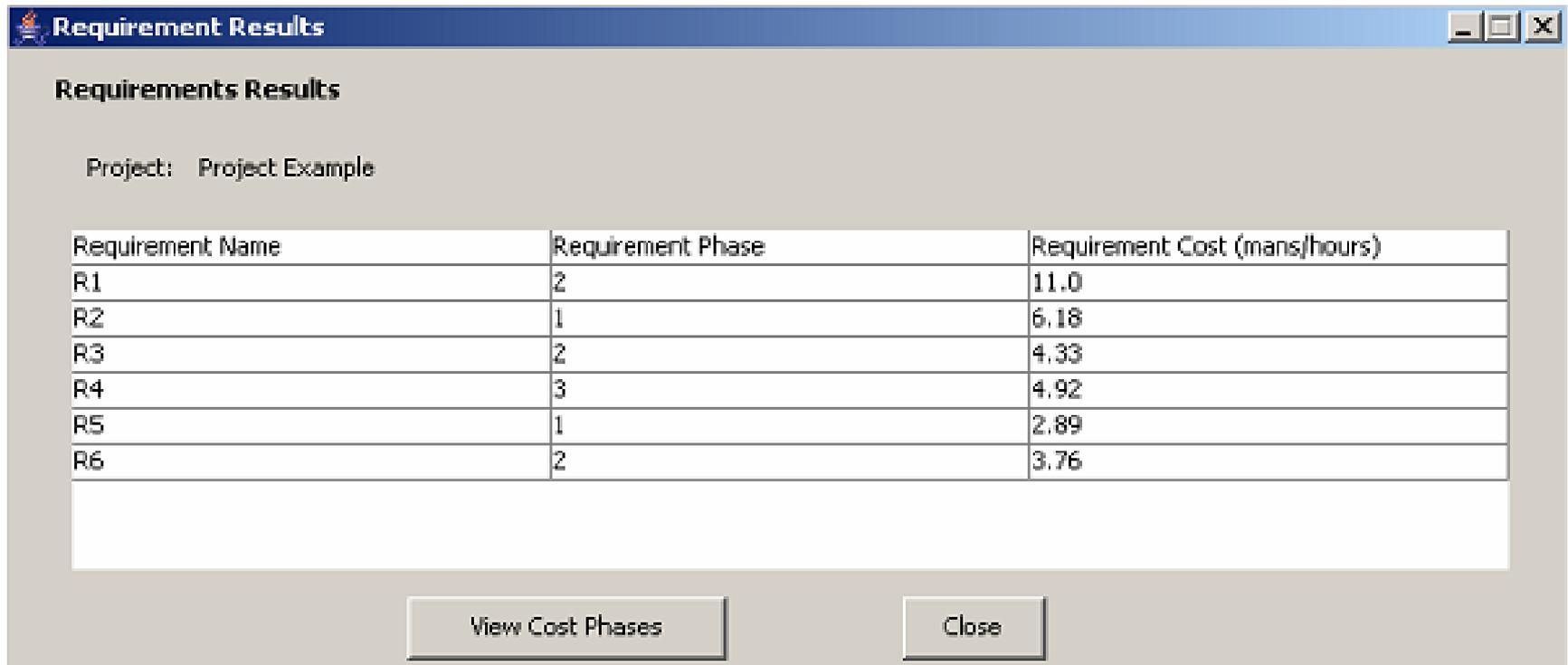
The central grid is titled 'Requisitos' and contains the following data:

RF x RNF	RNF001	RNF002	RNF003	RNF004	RNF005
RF001					
RF002					
RF003				✓	
RF004		✓			
RF005					
RF006					✓
RF007					✓
RF008					
RF009					

At the bottom of the window, there is a text area with the following content:

O sistema deverá cadastrar os tipos de requisitos desejados para o levantamento de requisitos
Utilizar controles ActiveX para acessar a interface do EA e manipular seus objetos

AspectCost



The screenshot shows a window titled "Requirement Results" with a blue header bar. Below the header, the text "Requirements Results" is displayed. Underneath, it says "Project: Project Example". A table with three columns is shown: "Requirement Name", "Requirement Phase", and "Requirement Cost (mans/hours)". The table contains six rows of data. At the bottom of the window, there are two buttons: "View Cost Phases" and "Close".

Requirement Name	Requirement Phase	Requirement Cost (mans/hours)
R1	2	11.0
R2	1	6.18
R3	2	4.33
R4	3	4.92
R5	1	2.89
R6	2	3.76

Requisitos

Requisitos Funcionais

- **RF01** - efetuar a exportação de um projeto especificado na ferramenta EA, disponibilizando estes dados em um arquivo no formato XML
- **RF.02** - interpretar o arquivo descrito no RF.01, buscando quais as dependências dos requisitos descritos
- **RF.03** - permitir ao usuário inserir anotações que identifiquem quais requisitos estão sendo atendidos nos códigos fontes, durante o desenvolvimento de um projeto
- **RF.04** - permitir ao usuário efetuar o mapeamento dos códigos fontes de um projeto, inserindo anotações de acordo com a interpretação do arquivo descrito no RF.01

Requisitos Funcionais

- **RF.05** - disponibilizar uma interface que mostre ao usuário quais as dependências (códigos fontes) associadas a um requisito do projeto
- **RF.06** - ponderar qual o custo que a alteração de um requisito representa para o projeto, através do processo AHP
- **RF.07** - disponibilizar uma interface que permita ao usuário consultar qual o custo que uma alteração pode causar no projeto
- **RF.08** - armazenar as operações de alteração dos códigos fontes e erros em um arquivo log, onde o usuário possa conferir a data e a classe que efetuou a operação

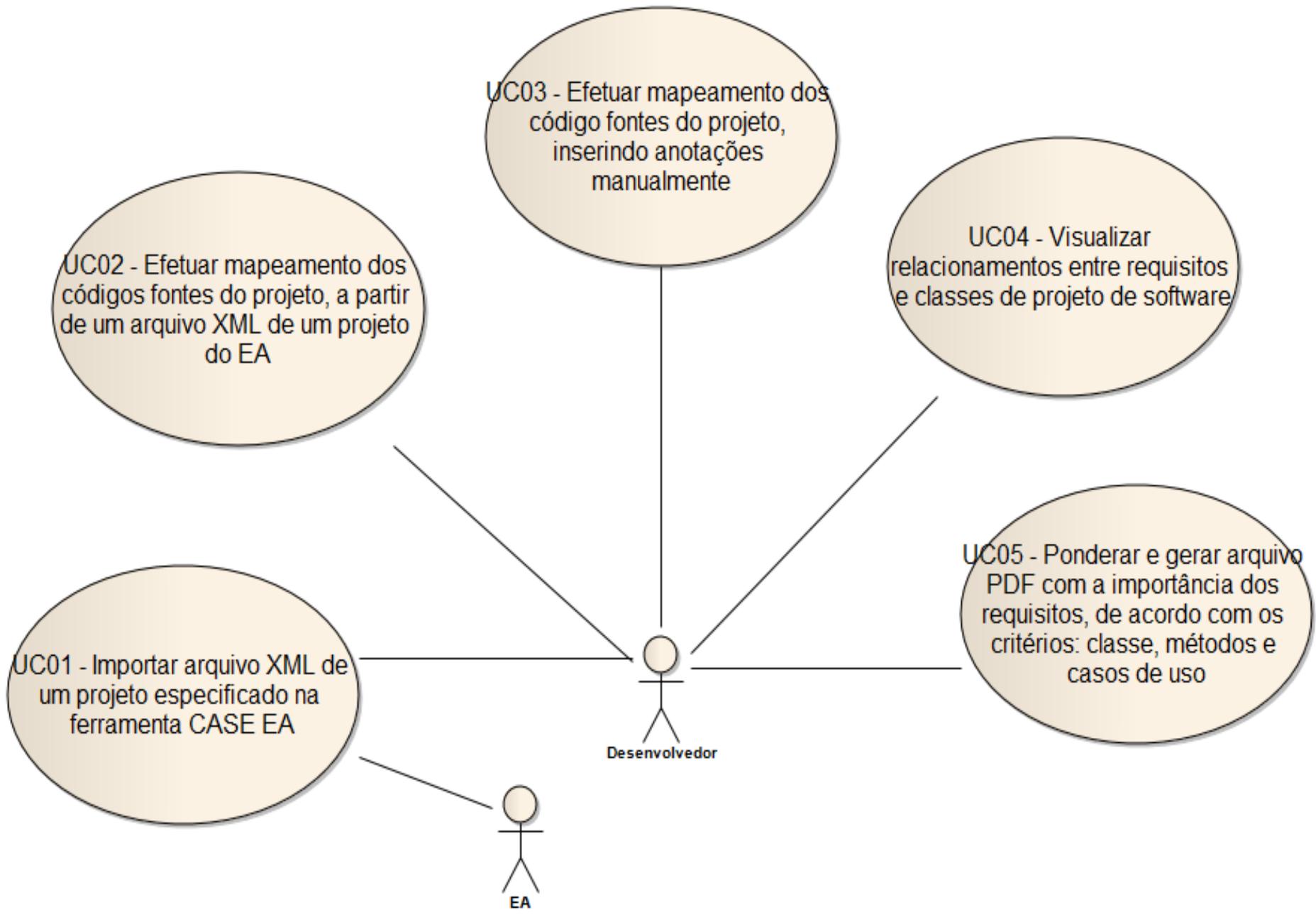
Requisitos não funcionais

- **RNF.01** - ser implementada utilizando o ambiente de desenvolvimento Eclipse 3.5
- **RNF.02** - ser implementada utilizando a linguagem de programação Java 1.6
- **RNF.03** - utilizar as bibliotecas Eaapi e SSJavaCom para a geração do arquivo XML
- **RNF.04** - utilizar a biblioteca JDOM para a leitura e manipulação do arquivo XML
- **RNF.05** - utilizar a biblioteca JasperReports 3.6.2 e IReport 3.6.2 para a geração do arquivo PDF com os custos dos requisitos.
- **RNF.06** - utilizar a biblioteca log4J na sua versão 1.2.16 para o armazenamento de logs

Especificação da solução

Casos de uso

Pacotes



Pacotes

- Principais pacotes:
 - parser.visitor
 - popup
 - view
 - services
 - mapper
 - report

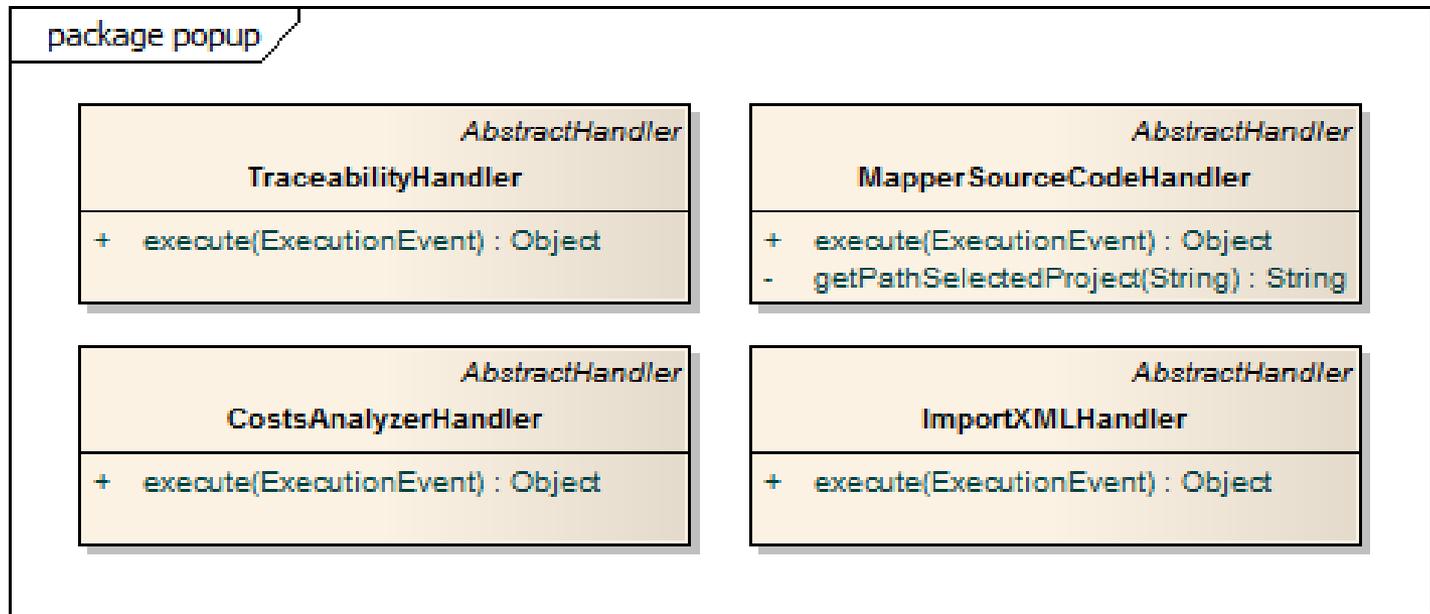
Pacote parser.visitor

package visitor

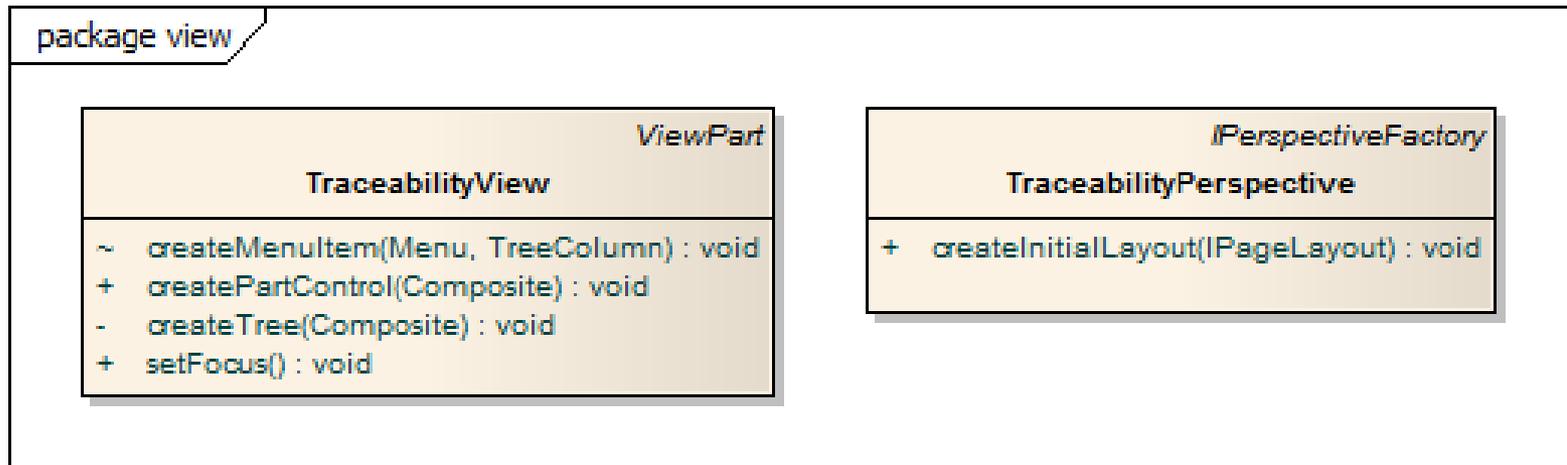
EARequisitManagerVisitor

- adjustString : String[]
- + EARequisitManagerVisitor(boolean, CompilationUnit, ICompilationUnit)
- + getClassDeclaration : List<NamedLineClass>
- + getIds : String[]
- + getLastDeclaredPackage : int
- + getMethodsAnnotations : Map<String, List<String>>
- hasAnnotation : boolean
- + visit(NormalAnnotation) : boolean
- + visit(ImportDeclaration) : boolean
- + visit(TypeDeclaration) : boolean
- + visit(MethodDeclaration) : boolean

Pacote popup



Pacote view



Pacote services

package services

CostAnalyzerService

- calculateAutoVector(double[][]): List<Double>
- + calculateCosts(double[][]): List<Double>
- calculateCostsOfRequirements(List<List<Double>>): List<Double>
- + calculateImportanceMatrix(List<String>, Map<String, List<RequirementsAssociations>>, TipoCriterio): double[]
- + calculateImportanceMatrix(List<RequirementsAssociations>): double[]
- + calculateImportanceMatrix(List<String>, List<Double>): double[]
- calculateNormalizedVector(double[][], List<Double>): List<List<Double>>
- + CostAnalyzerService()
- getClassesValues(List<String>, Map<String, List<RequirementsAssociations>>): List<Double>
- getMethodsValues(List<String>, Map<String, List<RequirementsAssociations>>): List<Double>
- roundValue(double, int): Double

ImportXMLService

- + exportFileXML(String, String): void
- getGUIDOfPackage(Repository): String
- + ImportXMLService()

GenerateMapperFromXML

- adjustMethodName(String): String
- + buildDocument(File): void
- + GenerateMapperFromXML()
- + getAllRequirements(): List<String>
- + getMapClassRequirements(): Map<String, List<String>>
- + requirementsAssociations(): List<RequirementsAssociations>

MapperSourceCodeService

- buildAnnotation(List<String>): String
- + doMapOnSourceCode(IProject, List<RequirementsAssociations>): void
- externalModify(ICompilationUnit): void
- getFilePath(IProject, ICompilationUnit): File
- getRequirementsFromClass(List<RequirementsAssociations>, String): List<String>
- getRequirementsFromMethod(List<RequirementsAssociations>, String, String): List<String>
- insertStatement(List<String>, String, int): void
- + MapperSourceCodeService()
- parse(ICompilationUnit): CompilationUnit
- saveModifications(File, List<String>): void
- sourceCodeToList(File): List<String>

TraceabilitySourceService

- getMethodsFromRequirement(Map<String, List<String>>, String): List<String>
- parse(ICompilationUnit): CompilationUnit
- + TraceabilitySourceService()
- + traceSources(IProject): Map<String, List<RequirementsAssociations>>

Pacote mapper

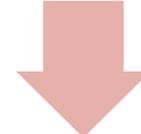
ElementVisitor

```
+ buildDocument() : void
- elementXMLToString(List<ElementXML>) : List<String>
+ ElementVisitor(File)
+ getAllRequirementsIds() : List<String>
- getNameOfElement(List<Element>) : String
- getValueFromAttribute(List<Attribute>, String) : String
- isAttributeIdEqualsClass(Element, String) : boolean
- isClassType(List<Element>) : boolean
- isElementId(List<Attribute>, String, String) : boolean
- isRequirementType(List<Element>) : boolean
- isTypeElement(List<Attribute>, TipoElemento) : boolean
- isUseCaseType(List<Element>) : boolean
+ searchClassByPackagedElement(String) : ElementXML
- searchClassByPackagedElement(Element, ElementXML, String) : void
+ searchClassesConectors(String) : List<ElementXML>
- searchClassesConectors(Element, List<ElementXML>, String) : void
+ searchClassOnXML() : List<ElementXML>
+ searchDependenciesOnXML(String) : List<ElementXML>
- searchElementOnXML(Element, List<ElementXML>, TipoElemento, String, String) : void
- searchMessages(Element, List<ElementXML>) : void
+ searchRequirementDescription(List<ElementXML>) : List<ElementXML>
+ searchRequirementsConectors(String) : List<String>
- searchRequirementsConectors(Element, List<String>, String) : void
+ searchRequirementsOnXML() : List<ElementXML>
+ searchUseCaseConectors(String) : List<ElementXML>
- searchUseCaseConectors(Element, List<ElementXML>, String) : void
- searchUseCaseMessage(Element, List<ElementXML>, String) : void
+ searchUseCaseMessages(String) : List<ElementXML>
+ searchUseCasesOnXML() : List<ElementXML>
+ visitElements() : void
```

RequirementsAssociations

```
+ addClass(String) : void
+ addMethod(String, String) : void
+ addUseCase(String) : void
+ existClass(String) : boolean
+ getClassesMethods() : Map<String, List<String>>
+ getMethods(String) : List<String>
+ getRequirement() : String
+ getSizeClasses() : int
+ getSizeMethods() : int
+ getSizeUseCases() : int
+ getUseCases() : List<String>
+ RequirementsAssociations()
+ setClassesMethods(Map<String, List<String>>) : void
+ setRequirement(String) : void
+ setUseCases(List<String>) : void
```

Pacote mapper



Pacote report

class report

WorkCollection

- + doCollection() : Collection<Object>
- getListCriterion(List<String>, List<Double>) : List<CostsBean>
- + WorkCollection(Map<String, List<RequirementsAssociations>>, GenerateMapperFromXML)

ListCostBean

- + addBean(CostsBean) : void
- + getCostsBeans() : List<CostsBean>
- + getNameCriterion() : String
- + setCostsBeans(List<CostsBean>) : void
- + setNameCriterion(String) : void

CostsBean

- + CostsBean()
- + getRequirement() : String
- + getValue() : Double
- + setRequirement(String) : void
- + setValue(Double) : void

ReportMaker

- + makeReport(Collection<Object>, String) : void
- + ReportMaker()

Implementação

Técnicas e ferramentas da implementação

Protótipo

Implementação

- Linguagem de programação: Java
- Ambiente de desenvolvimento: Eclipse
- Interface gráfica: SWT
- Bibliotecas:
 - **eaapi.jar** e **SSJavaCom**: integração com o EA
 - **jdom.jar**: leitura e interpretação do arquivo XML
 - **org.eclipse.jdt.core_3.5.2.jar**: parser de arquivos fontes
 - **jasperreports-3.6.2**: geração do arquivo PDF
 - **log4j**: armazenamento de log da aplicação

Protótipo

EA Requisite Manager - Custos dos Requisitos

Critério de Classes

Requisitos	Valor (%)
RF.01	27.16
RF.02	13.58
RF.04	13.58
RF.05	13.58
RF.06	13.58
RF.07	13.58
RF.08	4.93

Critério de Métodos

Requisitos	Valor (%)
RF.01	40.89
RF.02	12.23
RF.04	12.23
RF.05	10.56
RF.06	9.88
RF.07	9.88
RF.08	4.29

Critério de Casos de Uso

Requisitos	Valor (%)
RF.01	16.12
RF.02	16.12
RF.04	16.12
RF.05	16.12
RF.06	16.12
RF.07	16.12
RF.08	3.22

@Requi
pt

ect) {

Resultados e discussões

Resultados e discussões

	TraceFact-In	IRequirement	AspectCost	EA Requisite Manager
Rastreabilidade de requisitos	Sim	Sim	Não	Sim
Integração com ferramenta CASE	Requisite Pro	EA	StarUML e Rational Rose	EA
Finalidade de uso	Acadêmico	Acadêmico	Acadêmico	Acadêmico
Aplicação de método para quantificação	Não	Não	Sim	Sim
Banco de dados	MS SQL Server	-	XML	-
Análise de impacto com a alteração de requisitos	Sim	Sim	Sim	Sim
Linguagem utilizada para desenvolvimento	C#	Borland Delphi 6	Java	Java
Integração com IDE	Visual Studio	-	-	Eclipse

Resultados e discussões

- Utilização da AST para analisar o código fonte
- Interpretação de um arquivo XML do EA
- Importância e relevância da fase de projetos

Conclusão

Conclusão

- Resultados:
 - O objetivo deste trabalho foi alcançado
 - Os objetivos específicos foram alcançados
 - A ferramenta desenvolvida auxilia na automatização do processo de rastreabilidade e análise de impacto
 - A aplicação da metodologia AHP foi importante para se estabelecer como os artefatos (classes, casos de uso e requisitos) influenciam no tamanho dos requisitos de um projeto

Conclusão

- Resultados:
 - Leitura e interpretação dos dados do arquivo XML foram essenciais
 - Utilizar a técnica de interpretação com o auxílio de uma AST facilitou a inserção de anotações nos códigos fontes

Conclusão

- Algumas extensões:
 - disponibilizar uma interface em que o usuário consiga acompanhar o passo a passo da quantificação de requisitos
 - melhorar e aperfeiçoar o mecanismo de inserção de anotações nos códigos fontes, utilizando para isso, bibliotecas da IDE Eclipse
 - criar um mecanismo extensível, que permita a integração do *plugin* com outras ferramentas CASE de modelagem de projetos

Obrigado!
